
HTTP and HTTPS traffic:

HTTP:

A few things to keep in mind about HTTP:

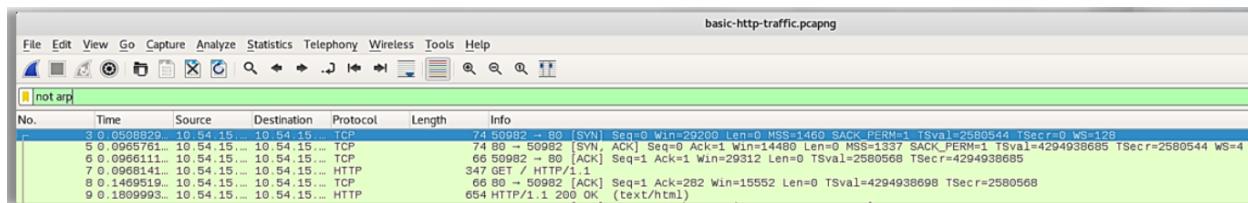
- Its client server protocol and the Client will request and the Server will respond
- HTTP responses include a 3 digit status code:
- HTTP messages include a message Header and Body
- HTTP uses methods to perform different actions like GET, POST, PUT, DELETE, OPTION, HEAD, PATCH.
- Not all METHODS are allowed to be used, mean all method are not permitted.

Some facts to distinguish Normal and Suspicious HTTP traffic:

Normal HTTP Traffic	Suspicious HTTP Traffic
Port 80, TCP Port 8080, TCP (used as alternate) Port 8088, TCP (used as alternate)	Malicious binaries (backdoors), scripts, web shells, etc. will use this port because typically, in all corporate environments, the port is open.
Plaintext traffic	If the traffic is encrypted, then most likely it's being used for malicious traffic. Malicious traffic can be in plaintext as well.
Web server typically in FQDN format.	The server will point to an IP address instead of FQDN format.

Since port 80,443 are open in all environment so the workstation can connect to websites in organization so they can do their job and this can be used in malicious way. If we can't see the data transmitted in the HTTP packet mean its encrypted then it's also suspicious. If we see a host is communicating over HTTP but its going to an IP not a FQDN or domain.

HTTP Normal Traffic:



Summary of the above packets:

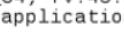
- We are seeing 6 packets which 4 of them related to TCP and 2 of them to HTTP
 - Packet 3,5,6 is the 3 way handshake, mean the HTTP protocol is sending its data over TCP and TCP connection starts with 3 way handshake.
 - In packet 7 we see the HTTP method GET.
 - In packet 9 we see the HTTP response code 200 OK.

Lets look at the first packet information:

- We see the Ethernet: which is the source/Destination MAC address
 - We see the Internet Protocol which shows the SRC/DEST IPs
 - We see the TCP, which shows the SRC/DEST Ports.
 - And we see the HTTP

So far we see the HTTP traffic is over DEST port 80, but it can be any port but the port 80 is the standard port for HTTP and seeing traffic over another port can be suspicious.

Lets look at the HTTP section of this packet:

```
▶ Frame 7: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface 0
▶ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: VMware_a1:61:66 (00:50:56:a1:61:66)
▶ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
▶ Transmission Control Protocol, Src Port: 50982, Dst Port: 80, Seq: 1, Ack: 1, Len: 281
▶ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    Host: 10.54.15.68\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
  \r\n
[Full request URI: http://10.54.15.68/] 
[HTTP request 1/2]
[Response in frame: 9]
[Next request in frame: 11]
```

We see in the first Line the HTTP Method, and we see in the HOST an IP not FQDN mean not the Domain name, as we know direct connection to IPs should be suspicious, but here its pointing to an Internal IP and that's a normal behavior as we might don't have Internal DNS servers.

From the bottom 2 line we see “Response in frame: 9” which means the response to this GET request in this packet 7 is in packet 9 and if we click it will take us to the packet 9 and lets do that.

```

▶ Frame 9: 654 bytes on wire (5232 bits), 654 bytes captured (5232 bits) on interface 0
▶ Ethernet II, Src: VMware_AutoIP (00:50:56:a1:f4:d0), Dst: 26:11:59:88:53:02 (26:11:59:88:53:02)
▶ Internet Protocol Version 4, Src: 10.54.15.68, Dst: 10.54.15.100
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 50982, Seq: 1, Ack: 282, Len: 588
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    Date: Tue, 23 May 2017 17:23:14 GMT\r\n
    Server: Apache/2.2.22 (Debian)\r\n
    X-Powered-By: PHP/5.4.4-14+deb7u14\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
  ▶ Content-Length: 315\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html\r\n
  \r\n
  [HTTP response 1/2]
  [Time since request: 0.084185252 seconds]
  ▶ Request in frame: 7
  ▶ Next request in frame: 11
  ▶ Next response in frame: 12
  Content-encoded entity body (gzip): 315 bytes -> 546 bytes
  File Data: 546 bytes
▼ Line-based text data: text/html
  <!DOCTYPE html>\n<html>\n
```

And in here we can do the same Jumping by clicking the “Request in frame: 7” which will take us back to the Request packet of this response packet.

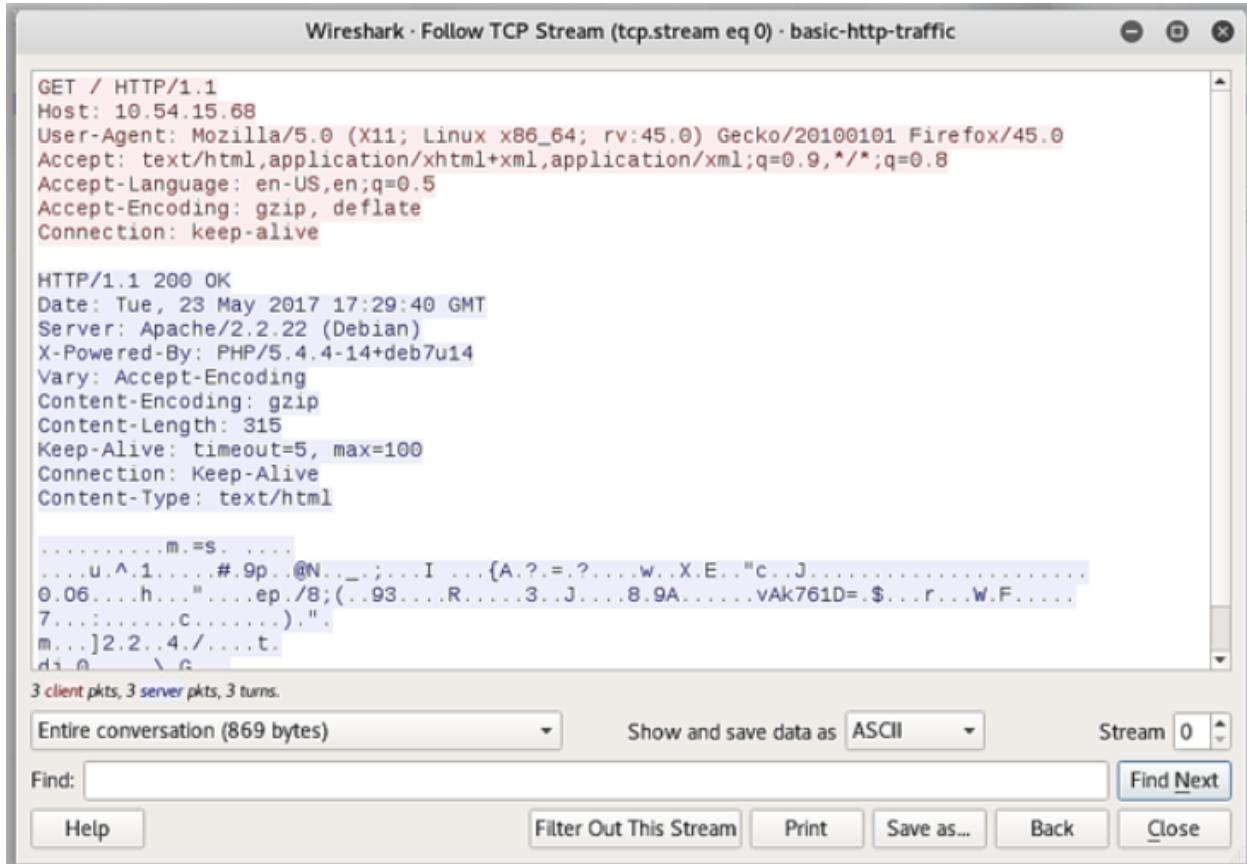
At the bottom we see the “Line-based test data: text/html” here it tell us what kind data we got in response and it will also show the data it self, which is text/html and lets look at it:

```

▼ Line-based text data: text/html
  <!DOCTYPE html>\n<html>\n  \n
  <head>\n    \n
    <meta charset="UTF-8">\n
    \n
    <title>Log-in - login.labs</title>\n
    \n
    <link rel="stylesheet" href="css/style.css" media="screen" type="text/css" />\n
  </head>\n  \n
  <body>\n    \n
    <div class="login-card">\n      <h1>Log-in</h1><br>\n      <form method="post" action="login.php">\n        <input type="text" name="user" autocomplete="off" placeholder="Username">\n        <input type="password" name="pass" placeholder="Password">\n        <input type="submit" name="login" class="login login-submit" value="login">\n      </form>\n    </div>\n  </body>\n</html>\n
```

Here we see the HTML which looks like the Login Page.

The same information can be obtained by right clicking the packet then click Follow > Follow TCP Stream, no matter we click the Respond or the Request, it will show both the data sent and the receive.



The red is the Data we sent, so like if it was POST request like logging, we would see it in Red color which is what we sent and the Blue is the response we got which we see the header and the data is in Blue.

NOTE: HTTP is clear text protocol that mean there is no encryption so if this traffic is intercepted then it will be readable.

If want to display the streams, then we can use this filter:

tcp.stream eq 0						
No.	Time	Source	Destination	Protocol	Length	Info
3	0.050982051	10.54.15.100	10.54.15.68	TCP	74 50982 → 80 [SYN] Seq=0	
5	0.0965611102	10.54.15.100	10.54.15.68	TCP	74 50982 → 80 [SYN, ACK] Seq=1	
6	0.0965611102	10.54.15.100	10.54.15.68	TCP	74 50982 → 80 [ACK] Seq=1	
7	0.0965611102	10.54.15.100	10.54.15.68	TCP	74 50982 → 80 [ACK] Seq=1	
8	0.1469519294	10.54.15.68	10.54.15.100	TCP	66 80 → 50982 [ACK] Seq=1	
9	0.180995372	10.54.15.68	10.54.15.100	HTTP	654 HTTP/1.1 200 OK (text)	
10	0.180995372	10.54.15.100	10.54.15.68	TCP	66 50982 → 80 [ACK] Seq=1	
11	0.212109429	10.54.15.100	10.54.15.68	HTTP	242 GET /favicon.ico HTTP/1.1	
12	0.296512761	10.54.15.68	10.54.15.100	HTTP	1242 HTTP/1.1 200 OK (text)	
13	0.327385188	10.54.15.100	10.54.15.68	TCP	66 50982 → 80 [ACK] Seq=56	
14	0.327385188	10.54.15.100	10.54.15.68	TCP	66 50982 → 80 [ACK] Seq=56	
15	0.290130066	10.54.15.100	10.54.15.68	TCP	66 50982 → 80 [FIN, ACK] Seq=56	
16	0.335866407	10.54.15.68	10.54.15.100	TCP	66 80 → 50982 [ACK] Seq=17	

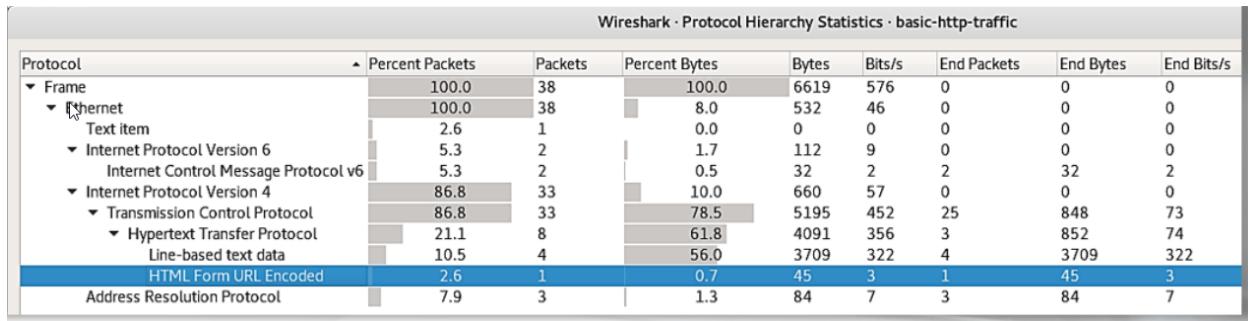
tcp.stream eq 2						
No.	Time	Source	Destination	Protocol	Length	Info
28	42.9173985920	10.54.15.100	10.54.15.68	TCP	74 51012 → 80 [SYN] Seq=0	
29	42.969140002	10.54.15.100	10.54.15.68	TCP	74 51012 → 80 [ACK] Seq=1	
30	42.969140002	10.54.15.100	10.54.15.68	TCP	66 51012 → 80 [ACK] Seq=1	
31	42.969353931	10.54.15.100	10.54.15.68	HTTP	501 POST /login.php HTTP/1.1	
32	43.016754695	10.54.15.68	10.54.15.100	TCP	66 80 → 51012 [ACK] Seq=1	
33	43.033810817	10.54.15.68	10.54.15.100	HTTP	604 HTTP/1.1 200 OK (text)	
34	43.033823724	10.54.15.100	10.54.15.68	TCP	66 51012 → 80 [ACK] Seq=43	
35	48.040277295	10.54.15.68	10.54.15.100	TCP	66 80 → 51012 [FIN, ACK] Seq=43	
36	48.040506072	10.54.15.100	10.54.15.68	TCP	66 51012 → 80 [FIN, ACK] Seq=43	
37	48.088082922	10.54.15.68	10.54.15.100	TCP	66 80 → 51012 [ACK] Seq=56	

tcp.stream eq 1						
No.	Time	Source	Destination	Protocol	Length	Info
18	20.862049221	10.54.15.100	10.54.15.68	TCP	74 51008 → 80 [SYN] Seq=0	
19	20.907960258	10.54.15.68	10.54.15.100	TCP	74 80 → 51008 [SYN, ACK] Seq=1	
20	20.907989940	10.54.15.100	10.54.15.68	TCP	66 51008 → 80 [ACK] Seq=1	
21	20.908172432	10.54.15.100	10.54.15.68	HTTP	358 GET /favicon.ico HTTP/1.1	
22	20.908172432	10.54.15.68	10.54.15.100	TCP	66 51008 → 51008 [ACK] Seq=1	
23	20.957862825	10.54.15.68	10.54.15.100	HTTP	568 HTTP/1.1 404 Not Found	
24	20.957877026	10.54.15.100	10.54.15.68	TCP	66 51008 → 80 [ACK] Seq=2	
25	25.062682870	10.54.15.68	10.54.15.100	TCP	66 80 → 51008 [FIN, ACK] Seq=2	
26	25.062789254	10.54.15.100	10.54.15.68	TCP	66 51008 → 80 [FIN, ACK] Seq=2	
27	26.008812475	10.54.15.68	10.54.15.100	TCP	66 80 → 51008 [ACK] Seq=5	

tcp.stream eq 3						
No.	Time	Source	Destination	Protocol	Length	Info

Protocol Hierarchy Statistics:

Here we can see the protocols that are within the PCAP file we looking at:



It shows the protocols based on the OSI layer kind, like we see the Frame, Ethernet, IPv6, IPv4, HTTP, ... here we see how many packet falls under which protocol as well. Like the IPv4 has 86.8 % of the packets which are 33 packets and then we have HTTP which is 21.1% and its 8 packets.

Export HTML Objects:

With this feature of Wireshark, we can save the objects like if an Image, a Binary, script is sent via HTTP, we can retrieve it like this.

The figure shows a Wireshark interface displaying a table titled "Wireshark · Export · HTTP object list". The table lists five saved HTML objects, each with a unique ID, the host name, content type, size, and filename. The objects include a main HTML page, a CSS file, a favicon, a login PHP script, and another login PHP script.

Packet	Hostname	Content Type	Size	Filename
9	10.54.15.68	text/html	546 bytes	/
12	10.54.15.68	text/css	2369 bytes	style.css
23	10.54.15.68	text/html	286 bytes	favicon.ico
31	10.54.15.68	application/x-www-form-urlencoded	45 bytes	login.php
33	10.54.15.68	text/html	508 bytes	login.php

Suspicious HTTP traffic:

We have to remember:

- The typical port of HTTP is 80

- Its clear text protocol
- Hosts are accessed using FQDN instead of IP addresses

Lets look at this what look suspicious to us?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.124.211.96	10.124.211.96	TCP	74	33020 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125613 TSecr=0 WS=128
2	0.056720	10.124.211.96	10.124.211.96	TCP	74	80 - 33020 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=222206 TSecr=125613 WS=4
3	0.056747	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1 Ack=1 Win=0 TSval=120627 TSecr=222206
4	0.056824	10.124.211.96	10.124.211.96	HTTP	349	GET / HTTP/1.1
5	0.133531	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 [ACK] Seq=1 Ack=284 Win=15552 Len=0 TSval=222223 TSecr=125627
6	0.133549	10.124.211.96	10.124.211.96	HTTP	101	[TCP Previous segment not captured] Continuation
7	0.133550	10.124.211.96	10.124.211.96	TCP	1301	[TCP Window Update] 33020 - 80 [ACK] Seq=284 Ack=1 Win=15552 Len=0 TSval=125627 TSecr=222223 SLE=1326 SRE=1361
8	0.156269	10.124.211.96	10.124.211.96	TCP	1301	[TCP Window Update] 33020 - 80 [ACK] Seq=284 Ack=1 Win=15552 Len=0 TSval=125627 TSecr=222223 TSecr=125627
9	0.156259	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1281 Win=33209 Len=0 TSval=125652 TSecr=222224
10	3.481485	10.124.211.96	10.124.211.96	HTTP	389	GET /news.php HTTP/1.1
11	3.552272	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
12	3.552278	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=0 Win=2868 Len=36096 Len=0 TSval=126501 TSecr=223083
13	4.552589	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
14	5.968993	10.124.211.96	10.124.211.96	HTTP	410	GET /newsdetails.php?id=26 HTTP/1.1
15	6.020430	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
16	6.020431	10.124.211.96	10.124.211.96	HTTP	66	33020 - 80 [ACK] Seq=951 Ack=5298 Win=41856 Len=0 TSval=127118 TSecr=223701
17	6.020437	10.124.211.96	10.124.211.96	TCP	89	HTTP/1.1 200 OK (text/html)
18	6.020440	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=951 Ack=5295 Win=41856 Len=0 TSval=127118 TSecr=223701
19	6.020471	10.124.211.96	10.124.211.96	HTTP	373	GET /newsdetails.php?id=26%27 HTTP/1.1
20	6.020476	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=951 Ack=5295 Win=41856 Len=0 TSval=127118 TSecr=223701
21	9.517192	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
22	9.517210	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1284 Win=6640 Len=44800 Len=0 TSval=127992 TSecr=224575
23	9.517216	10.124.211.96	10.124.211.96	HTTP	68	HTTP/1.1 200 OK (text/html)
24	9.517218	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1284 Ack=6642 Win=44800 Len=0 TSval=127992 TSecr=224575
25	14.520484	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 FIN, ACK Seq=6642 Ack=1258 Win=18768 Len=0 TSval=125826 TSecr=127992
26	14.520662	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [FIN, ACK] Seq=1288 Ack=6643 Win=44800 Len=0 TSval=129243 TSecr=225826
27	14.582082	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 [ACK] Seq=6643 Ack=1259 Win=18768 Len=0 TSval=125824 TSecr=129243
28	17.437742	10.124.211.96	10.124.211.96	HTTP	74	33022 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=12973 TSecr=0 WS=128
29	17.503661	10.124.211.96	10.124.211.96	TCP	74	80 - 33022 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=226572 TSecr=129973 WS=4

Here we see in the packet 20 there is GET request to /newsdetials.php?id=26%27 which this %27 is url encoding:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.124.211.96	10.124.211.96	TCP	74	33020 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125613 TSecr=0 WS=128
2	0.056720	10.124.211.96	10.124.211.96	TCP	74	80 - 33020 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=222206 TSecr=125613 WS=4
3	0.056747	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1 Ack=1 Win=0 TSval=120627 TSecr=222206
4	0.056824	10.124.211.96	10.124.211.96	HTTP	349	GET / HTTP/1.1
5	0.133531	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 [ACK] Seq=1 Ack=284 Win=15552 Len=0 TSval=222223 TSecr=125627
6	0.133549	10.124.211.96	10.124.211.96	HTTP	101	[TCP Previous segment not captured] Continuation
7	0.133550	10.124.211.96	10.124.211.96	TCP	1301	[TCP Window Update] 33020 - 80 [ACK] Seq=284 Ack=1 Win=15552 Len=0 TSval=125647 TSecr=222223 SLE=1326 SRE=1361
8	0.156269	10.124.211.96	10.124.211.96	TCP	1301	[TCP Window Update] 33020 - 80 [ACK] Seq=284 Ack=1 Win=15552 Len=0 TSval=125627 TSecr=222224
9	0.156259	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1281 Win=44800 Len=0 TSval=127992 TSecr=224575
10	3.481485	10.124.211.96	10.124.211.96	HTTP	410	GET /news.php HTTP/1.1
11	3.552272	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
12	3.552305	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=0 Ack=2868 Win=36096 Len=0 TSval=126501 TSecr=223083
13	3.552578	10.124.211.96	10.124.211.96	TCP	1353	HTTP/1.1 200 OK (text/html)
14	6.020430	10.124.211.96	10.124.211.96	HTTP	66	33020 - 80 [ACK] Seq=951 Ack=3973 Win=39040 Len=0 TSval=126501 TSecr=223083
15	6.020437	10.124.211.96	10.124.211.96	TCP	410	HTTP/1.1 200 OK (text/html)
16	6.020439	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
17	6.020461	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=951 Ack=5299 Win=41856 Len=0 TSval=127118 TSecr=223701
18	6.020470	10.124.211.96	10.124.211.96	HTTP	83	HTTP/1.1 200 OK (text/html)
19	6.020471	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=951 Ack=5295 Win=41856 Len=0 TSval=127118 TSecr=223701
20	6.020476	10.124.211.96	10.124.211.96	HTTP	373	GET /newsdetials.php?id=26%27 HTTP/1.1
21	9.517192	10.124.211.96	10.124.211.96	TCP	1391	[TCP segment of a reassembled PDU]
22	9.517210	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1284 Ack=6640 Win=44800 Len=0 TSval=127992 TSecr=224575
23	9.517218	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [ACK] Seq=1284 Ack=6642 Win=44800 Len=0 TSval=127992 TSecr=224575
24	14.520484	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 FIN, ACK Seq=6642 Ack=1258 Win=18768 Len=0 TSval=125826 TSecr=127992
25	14.520662	10.124.211.96	10.124.211.96	TCP	66	33020 - 80 [FIN, ACK] Seq=1288 Ack=6643 Win=44800 Len=0 TSval=129243 TSecr=225826
26	14.582082	10.124.211.96	10.124.211.96	TCP	66	80 - 33020 [ACK] Seq=6643 Ack=1259 Win=18768 Len=0 TSval=125824 TSecr=129243
27	17.437742	10.124.211.96	10.124.211.96	HTTP	74	33022 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=12973 TSecr=0 WS=128
28	17.503661	10.124.211.96	10.124.211.96	TCP	74	80 - 33022 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=226572 TSecr=129973 WS=4

Here is highlighted that part, and that %27 means a single quote ('), this is used by attackers to find SQL injection. This might be a Type but for this to prove we need to look to dive deep and prov it.

If look further we see the signs of this SQLi inject in more packets as well.

No.	Time	Source	Destination	Protocol	Length	Info
29	9.517192	10.124.211.96	10.124.211.96	HTTP	1302	/newsdetails.php?id=26%27 HTTP/1.1
21	9.517192	10.124.211.96	10.124.211.96	TCP	1302 [TCP segment of a reassembled PDU]	
22	9.517210	10.124.211.96	10.124.211.96	TCP	66 33020 → 80 [ACK] Seq=1258 Ack=6640 Win=44800 Len=0 TSeqval=127992 TSecr=224575	
23	9.517216	10.124.211.96	10.124.211.96	HTTP	68 HTTP/1.1 200 OK (text/html)	
24	9.517218	10.124.211.96	10.124.211.96	TCP	66 33020 → 80 [ACK] Seq=1258 Ack=6642 Win=44800 Len=0 TSeqval=127992 TSecr=224575	
25	14.520484	10.124.211.96	10.124.211.96	TCP	66 80 → 33020 [FIN, ACK] Seq=6642 Ack=1258 Win=18768 Len=0 TSeqval=225826 TSecr=127992	
26	14.520662	10.124.211.96	10.124.211.96	TCP	66 33020 → 80 [FIN, ACK] Seq=1258 Ack=6643 Win=44800 Len=0 TSeqval=129243 TSecr=225826	
27	14.582082	10.124.211.96	10.124.211.96	TCP	66 80 → 33020 [ACK] Seq=6644 Ack=1259 Win=18769 Len=0 TSeqval=225842 TSecr=129243	
28	17.437742	10.124.211.96	10.124.211.96	TCP	74 33022 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSeqval=129973 TSecr=0 WS=128	
29	17.437742	10.124.211.96	10.124.211.96	TCP	74 33022 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSeqval=129973 SACK_PERM=1 TSeqval=226572 TSecr=129973 WS=4	
30	17.503697	10.124.211.96	10.124.211.96	TCP	66 33022 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSeqval=129980 TSecr=226572	
31	17.504112	10.124.211.96	10.124.211.96	HTTP	389 GET /newsdetails.php?id=26%20and%201;--%20- HTTP/1.1	
32	17.575934	10.124.211.96	10.124.211.96	TCP	66 80 → 33022 [ACK] Seq=1 Ack=1 Win=15552 Len=0 TSeqval=226590 TSecr=129999	
33	17.578773	10.124.211.96	10.124.211.96	TCP	84 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]	
34	17.578787	10.124.211.96	10.124.211.96	TCP	78 [TCP Window Update] 33022 → 80 [ACK] Seq=324 Ack=1 Win=30336 Len=0 TSeqval=130008 TSecr=226590 SLE=1320 SRE=1344	
35	17.579068	10.124.211.96	10.124.211.96	TCP	1391 [TCP Out-of-Order] 33022 → 33022 [ACK] Seq=1 Ack=324 Win=15552 Len=1325 TSeqval=226591 TSecr=129999	
36	17.579088	10.124.211.96	10.124.211.96	TCP	66 33022 → 80 [ACK] Seq=324 Ack=1344 Win=33280 Len=0 TSeqval=130008 TSecr=226591	
37	22.579488	10.124.211.96	10.124.211.96	TCP	66 33022 → 80 [FIN, ACK] Seq=324 Ack=1344 Win=33280 Len=0 TSeqval=131258 TSecr=226591	
38	22.581438	10.124.211.96	10.124.211.96	TCP	66 80 → 33022 [ACK] Seq=1344 Ack=325 Win=15552 Len=0 TSeqval=131258 TSecr=226591	
39	22.584144	10.124.211.96	10.124.211.96	TCP	66 33022 → 80 [ACK] Seq=325 Ack=1345 Win=33280 Len=0 TSeqval=131259 TSecr=227842	
40	22.625046	10.124.211.96	10.124.211.96	TCP	66 80 → 33022 [ACK] Seq=1345 Ack=325 Win=15552 Len=0 TSeqval=131258 TSecr=227842	
41	25.101234	10.124.211.96	10.124.211.96	TCP	74 33024 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSeqval=131888 TSecr=0 WS=128	
42	25.141132	10.124.211.96	10.124.211.96	TCP	74 80 → 33024 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSeqval=228481 TSecr=131888 WS=4	
43	25.141157	10.124.211.96	10.124.211.96	TCP	66 33024 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSeqval=131899 TSecr=228481	
44	25.141361	10.124.211.96	10.124.211.96	HTTP	389 GET /newsdetails.php?id=26%20and%201;--%20- HTTP/1.1	
45	25.186826	10.124.211.96	10.124.211.96	TCP	66 80 → 33024 [ACK] Seq=1 Ack=324 Win=15552 Len=0 TSeqval=228492 TSecr=131899	
46	25.187157	10.124.211.96	10.124.211.96	HTTP	1288 HTTP/1.1 200 OK (text/html)	
47	25.187188	10.124.211.96	10.124.211.96	TCP	66 33024 → 80 [ACK] Seq=324 Ack=1220 Win=32128 Len=0 TSeqval=131910 TSecr=228493	
48	30.188296	10.124.211.96	10.124.211.96	TCP	66 33024 → 80 [FIN, ACK] Seq=324 Ack=1220 Win=32128 Len=0 TSeqval=133160 TSecr=228493	

Here If we look at packet 31,44 we see more attempts of using URL encoding which are Payloads of SQLi.

So its very important for a good hunter to understand Red Team tactics and recognize malicious actions, so we have to think like an attack.

Lets check if the attacker is using a tool or the attacker is trying it manually, and we do that by looking at the User-Agent which is the client or the browser that makes the request and this can be easily spoofed but we will still look at it:

```
Frame 20: 373 bytes on wire (2984 bits), 373 bytes captured (2984 bits)
Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
Transmission Control Protocol, Src Port: 33020, Dst Port: 80, Seq: 951, Ack: 5315, Len: 307
Hypertext Transfer Protocol
  > GET /newsdetails.php?id=26%27 HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
\r\n
[Full request URI: http://10.124.211.96/newsdetails.php?id=26%27]
[HTTP request 4/4]
[Prev request in frame: 15]
[Response in frame: 23]
```

We see the user agent is Mozilla and the OS is Linux and we said this can be easily spoofed, lets check for the other 2 packets, the 44 and 31:

```

▶ Frame 44: 389 bytes on wire (3112 bits), 389 bytes captured (3112 bits)
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33024, Dst Port: 80, Seq: 1, Ack: 1, Len: 323
▼ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=26%20and%201=2;--%20- HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
  \r\n
  [Full request URI: http://10.124.211.96/newsdetails.php?id=26%20and%201=2;--%20-]
  [HTTP request 1/1]
  [Response in frame: 46]

  observations = observations
  @control = control
  @candidates = observations - {control}

▶ Frame 31: 389 bytes on wire (3112 bits), 389 bytes captured (3112 bits)
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: Vmware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33022, Dst Port: 80, Seq: 1, Ack: 1, Len: 323
▼ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=26%20and%201=1;--%20- HTTP/1.1\r\n
    Host: 10.124.211.96\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
  \r\n
  [Full request URI: http://10.124.211.96/newsdetails.php?id=26%20and%201=1;--%20-]
  [HTTP request 1/1]

```

It looks the same User-agent, this might mean the attacker is doing this SQLi manually.

After we see an attack attempts we should always check if the attack was successful or not, so lets look at the packet 20 response which the attacker put %27 which is (').

```

<t!-- end sidebar -->\n<div id="content">\n<div></div>\n<div class="boxed">\n<h1 class="title2">\n<br />\n<b>Warning</b>: mysqli_fetch_array() expects parameter 1 to be mysqli_result, boolean given in <b>/var/www/newsdetails.php</b> on line <b>11</b><br />\n<br />\n<h1>\n<p>\n<br />\n<b>Warning</b>: mysqli_fetch_array() expects parameter 1 to be mysqli_result, boolean given in <b>/var/www/newsdetails.php</b> on line <b>29</b><br />\n</p>\n</div>

```

Here we see In the response the attacker got a mysql Error, that mean that our system is indeed vulnerable mean this single quote (') made to the Database was executed by the database.

The response to packet 31 isn't captured for some reason as we see it in black:

33 17.578773	10.124.211.96	10.124.211.200	TCP	84 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
34 17.578787	10.124.211.200	10.124.211.96	TCP	78 [TCP Window Update] 33022 - 80 [ACK] Seq=324 Ack=1 Win=30336 Len=0 TSeq=130008 TScr=226590 SLE=1326 SRE=1344
35 17.579068	10.124.211.200	10.124.211.96	TCP	1391 [TCP Out-Of-Order] 80 - 33022 [ACK] Seq=1 Ack=324 Win=15552 Len=325 TSeq=226591 TScr=129989
36 17.579088	10.124.211.200	10.124.211.96	TCP	88 33022 - 80 [ACK] Seq=224 Ack=1344 Win=22280 Len=0 TSeq=130008 TScr=226591

Lets look at the packet response of the packet 44 attempt of SQLi:

```

<t><t><t><form id="form1" method="post" action="login.php">\n
<t><t><t><fieldset>\n
<t><t><t><label for="inputtext1">Email:</label>\n
<t><t><t><input id="inputtext1" type="text" name="username" value="" />\n
<t><t><t><label for="inputtext2">Password:</label>\n
<t><t><t><input id="inputtext2" type="password" name="password" value="" />\n
<t><t><t><input id="inputsubmit1" type="submit" name="submit" value="Login" />\n
<t><t><p><a href="#">Forgot your password?</a><br />\n
<t><t><t></fieldset>\n
<t><t></form>\n
<t></div>\n
<t></div>\n
<t>!-- end sidebar -->\n
<t><div id="content">\n
<t><div></div>\n
<t><div class="boxed">\n
<t><t><h1 class="title2">\n
<n
<n
<n
</h1>\n
<p>\n
</p>\n
<n
</div>\n
<t></div>\n
<t>!-- end content -->\n
<t><div style="clear: both;">&nbsp;</div>\n
</div>\n
<!-- end page -->\n
<div id="footer">\n
<t><p id="legal">Copyright &copy; 2017 AwdMgmt.</p>\n
<t><p id="links"><a href="#">Privacy terms</a> | <a href="#">Licensing</a></p>\n
</div>

```

Even though we saw the SQLi attempt on packet 44, in its response in packet 46, we don't see any error.

One common thing is that when an attacker finds a vulnerability by manually looking for it, it will start to exploit it using tools, so here the attacker did discover the vulnerability.

Lets look at the packets after it:

No.	Time	Source	Destination	Protocol	Length	Info
56	34.163059	10.124.211.200	10.124.211.96	HTTP	266 GET /newsdetails.php?id=1 HTTP/1.1	
57	34.169322	10.124.211.96	10.124.211.200	TCP	74 80 - 33028 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TStamp=230739 TSecr=134145 WS=4	
58	34.169364	10.124.211.200	10.124.211.96	TCP	66 33028 - 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TStamp=134156 TSecr=230739	
59	34.180457	10.124.211.200	10.124.211.96	TCP	74 33030 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TStamp=134159 TSecr=0 WS=128	
60	34.202983	10.124.211.96	10.124.211.200	TCP	66 80 - 33028 [ACK] Seq=1 Ack=201 Win=15552 Len=0 TStamp=230747 TSecr=134154	
61	34.206414	10.124.211.96	10.124.211.200	TCP	1391 [TCP segment of a reassembled PDU]	
62	34.206422	10.124.211.96	10.124.211.200	TCP	66 33026 - 80 [ACK] Seq=201 Ack=1322 Win=32128 Len=0 TStamp=134165 TSecr=230748	
63	34.206495	10.124.211.96	10.124.211.200	HTTP	82 HTTP/1.1 200 OK (text/html)	
64	34.206501	10.124.211.200	10.124.211.96	TCP	66 33026 - 80 [ACK] Seq=201 Ack=1342 Win=32128 Len=0 TStamp=134165 TSecr=230748	
65	34.206508	10.124.211.96	10.124.211.200	TCP	66 80 - 33026 [FIN, ACK] Seq=1342 Ack=201 Win=15552 Len=0 TStamp=230748 TSecr=134154	
66	34.207131	10.124.211.200	10.124.211.96	TCP	66 33026 - 80 [FIN, ACK] Seq=201 Ack=1343 Win=32128 Len=0 TStamp=134165 TSecr=230748	
67	34.218945	10.124.211.96	10.124.211.200	TCP	74 80 - 33030 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TStamp=230751 TSecr=134158 WS=4	
68	34.218972	10.124.211.200	10.124.211.96	TCP	66 33030 - 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TStamp=134168 TSecr=230751	
69	34.230922	10.124.211.200	10.124.211.96	TCP	74 33032 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TStamp=134171 TSecr=0 WS=128	
70	34.230947	10.124.211.96	10.124.211.200	TCP	66 33032 - 80 [ACK] Seq=1 Ack=1 Win=15552 Len=0 TStamp=134171 TSecr=134165	
71	34.256905	10.124.211.96	10.124.211.200	TCP	74 80 - 33032 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TStamp=230764 TSecr=134171 WS=4	
72	34.269934	10.124.211.200	10.124.211.96	TCP	66 33032 - 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TStamp=134181 TSecr=230764	
73	35.126264	10.124.211.200	10.124.211.96	HTTP	266 GET /newsdetails.php?id=1 HTTP/1.1	
74	35.126569	10.124.211.200	10.124.211.96	TCP	74 33034 - 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TStamp=134399 TSecr=0 WS=128	
75	35.166226	10.124.211.96	10.124.211.200	TCP	66 80 - 33028 [ACK] Seq=1 Ack=201 Win=15552 Len=0 TStamp=230988 TSecr=134395	
76	35.166267	10.124.211.96	10.124.211.200	TCP	74 80 - 33034 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TStamp=230988 TSecr=134395 WS=4	
77	35.166941	10.124.211.200	10.124.211.96	TCP	66 33026 - 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TStamp=134405 TSecr=230988	
78	35.166996	10.124.211.96	10.124.211.200	TCP	1391 [TCP segment of a reassembled PDU]	
79	35.168935	10.124.211.200	10.124.211.96	TCP	66 33028 - 80 [ACK] Seq=201 Ack=1326 Win=32128 Len=0 TStamp=134405 TSecr=230988	
80	35.168926	10.124.211.96	10.124.211.200	HTTP	82 HTTP/1.1 200 OK (text/html)	
81	35.168943	10.124.211.200	10.124.211.96	TCP	66 33028 - 80 [ACK] Seq=201 Ack=1342 Win=32128 Len=0 TStamp=134405 TSecr=230988	
82	35.168962	10.124.211.96	10.124.211.200	TCP	66 80 - 33028 [FIN, ACK] Seq=1342 Ack=201 Win=15552 Len=0 TStamp=230988 TSecr=134395	
83	35.178154	10.124.211.200	10.124.211.96	TCP	66 33028 - 80 [FIN, ACK] Seq=201 Ack=1343 Win=32128 Len=0 TStamp=134406 TSecr=230988	

Strangely we don't see any SQLi payload in the ID field. Is it because the attacker found the vulnerability and stopped?

Lets take a closer look at the packet 56:

```

▶ Frame 56: 266 bytes on wire (2128 bits), 266 bytes captured (2128 bits)
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: VMware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33026, Dst Port: 80, Seq: 1, Ack: 1, Len: 200
▼ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=1 HTTP/1.1\r\n
    Accept-Encoding: gzip,deflate\r\n
    Host: 10.124.211.96\r\n
    Accept: */*\r\n
    User-Agent: sqlmap/1.1.4#stable (http://sqlmap.org)\r\n
    Connection: close\r\n
    Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://10.124.211.96/newsdetails.php?id=1]
  [HTTP request 1/1]
  [Response in frame: 63]

```

As we see the user agent is “sqlmap” which is a tool used to exploit SQLi vulnerability. As we guessed the attackers will find vulnerabilities and then use automated tools to exploit the best way they can.

The same thing apply to the rest of the packets from the same IP, all the request have “sqlmap” User-Agent.

If we scroll down, in packet 105 we will see that the ID field is target again:

No.	Time	Source	Destination	Protocol	Length	Info
84 35 206762	10.124.211.200	10.124.211.96	HTTP	269	GET /newsdetails.php?id=4616 HTTP/1.1	
85 35 209109	10.124.211.200	10.124.211.96	TCP	74	33036 - 80 [SYN] Seq:0 Win=32900 MSS=1460 SACK_PERM=1 TSeqval=134415 TSeqr=0 WS=128	
86 35 210593	10.124.211.96	10.124.211.200	TCP	66	80 - 33028 [ACK] Seq:1343 Ack+202 Win=15552 Len=0 TSeqval=230999 TSeqr=134406	
87 35 249943	10.124.211.96	10.124.211.200	TCP	66	80 - 33030 [ACK] Seq:1343 Ack+204 Win=15552 Len=0 TSeqval=231000 TSeqr=134415	
88 35 249959	10.124.211.96	10.124.211.200	TCP	66	80 - 33032 [ACK] Seq:1343 Ack+206 Win=15552 Len=0 TSeqval=231000 TSeqr=134415	
89 35 249969	10.124.211.96	10.124.211.200	TCP	66	80 - 33034 [ACK] Seq:1343 Ack+208 Win=15552 Len=0 TSeqval=231000 TSeqr=134415	
90 35 252937	10.124.211.96	10.124.211.200	TCP	66	33036 - 80 [ACK] Seq:1 Ack=1 Win=29312 Len=0 TSeqval=134424 TSeqr=231009	
91 35 252950	10.124.211.96	10.124.211.200	TCP	66	33030 - 80 [ACK] Seq:204 Ack+188 Win=32120 Len=0 TSeqval=134426 TSeqr=231009	
92 35 252977	10.124.211.96	10.124.211.200	TCP	66	80 - 33032 [ACK] Seq:204 Ack+188 Ack+204 Win=15552 Len=0 TSeqval=231009 TSeqr=134415	
93 35 253207	10.124.211.96	10.124.211.200	TCP	66	33030 - 80 [FIN, ACK] Seq:204 Ack+184 Win=32120 Len=0 TSeqval=134426 TSeqr=231009	
94 35 257209	10.124.211.96	10.124.211.200	HTTP	269	GET /newsdetails.php?id=1416 HTTP/1.1	
95 35 260419	10.124.211.96	10.124.211.200	TCP	74	33038 - 80 [SYN] Seq:0 Win=29200 MSS=1460 SACK_PERM=1 TSeqval=134428 TSeqr=0 WS=128	
96 35 294495	10.124.211.96	10.124.211.200	TCP	66	80 - 33030 [ACK] Seq:118 Ack+205 Win=15552 Len=0 TSeqval=231010 TSeqr=134426	
97 35 294538	10.124.211.96	10.124.211.200	TCP	66	80 - 33032 [ACK] Seq:118 Ack+206 Win=15552 Len=0 TSeqval=231011 TSeqr=134427	
98 35 299565	10.124.211.96	10.124.211.200	TCP	74	80 - 33038 [SYN, ACK] Seq:0 Ack:1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSeqval=231021 TSeqr=134428 WS=4	
99 35 299583	10.124.211.96	10.124.211.200	TCP	66	33038 - 80 [ACK] Seq:1 Ack=1 Win=29312 Len=0 TSeqval=134427 TSeqr=231021	
100 35 300920	10.124.211.96	10.124.211.200	TCP	66	[TCP Previous segment not captured]	
101 35 300926	10.124.211.96	10.124.211.200	TCP	66	33032 [FIN, ACK] Seq:204 Ack+188 Win=15552 Len=0 TSeqval=231021 SSeq=1183 RSeq=1184	
102 35 301309	10.124.211.96	10.124.211.200	TCP	1248	33032 [TCP segment of a retransmission]	
103 35 301315	10.124.211.96	10.124.211.200	TCP	66	33032 - 80 [ACK] Seq:204 Ack+184 Win=32128 Len=0 TSeqval=134439 TSeqr=231022	
104 35 301742	10.124.211.96	10.124.211.200	TCP	66	33032 - 80 [FIN, ACK] Seq:204 Ack+184 Win=32128 Len=0 TSeqval=134439 TSeqr=231022	
105 35 304677	10.124.211.96	10.124.211.200	HTTP	294	GET /newsdetails.php?id=1%27%2C..%2C%2C%2C%22.%2C HTTP/1.1	
106 35 343917	10.124.211.96	10.124.211.200	TCP	66	80 - 33030 [SYN] Seq:0 Win=29200 MSS=1460 SACK_PERM=1 TSeqval=134441 TSeqr=0 WS=128	
107 35 343916	10.124.211.96	10.124.211.200	TCP	66	80 - 33032 [ACK] Seq:1184 Ack+205 Win=15552 Len=0 TSeqval=231032 TSeqr=134439	
108 35 343937	10.124.211.96	10.124.211.200	TCP	66	80 - 33034 [ACK] Seq:1184 Ack+206 Win=15552 Len=0 TSeqval=231033 TSeqr=134439	
109 35 347304	10.124.211.96	10.124.211.200	HTTP	1357	HTTP/1.1 200 OK (text/html)	
110 35 347317	10.124.211.96	10.124.211.200	TCP	66	33034 - 80 [ACK] Seq:225 Ack:1292 Win=32128 Len=0 TSeqval=134450 TSeqr=231033	
111 35 347398	10.124.211.96	10.124.211.200	TCP	66	RST, ACK, Seq:1292 Len=0 Win=15552 TSeqval=231032 TSeqr=134427	

We see the URL encoding here, lets look into this packet 105:

```

▶ Frame 105: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
▶ Ethernet II, Src: 1a:3a:46:bf:43:91 (1a:3a:46:bf:43:91), Dst: VMware_a1:4e:f0 (00:50:56:a1:4e:f0)
▶ Internet Protocol Version 4, Src: 10.124.211.200, Dst: 10.124.211.96
▶ Transmission Control Protocol, Src Port: 33034, Dst Port: 80, Seq: 1, Ack: 1, Len: 224
▼ Hypertext Transfer Protocol
  ▶ GET /newsdetails.php?id=1%27%2C..%2C%2C%2C%22.%2C HTTP/1.1\r\n
    Request Method: GET
    Request URI: /newsdetails.php?id=1%27%2C..%2C%2C%2C%22.%2C
    Request URI Path: /newsdetails.php
    Request URI Query: id=1%27%2C..%2C%2C%2C%22.%2C
      Request URI Query Parameter: id=1%27%2C..%2C%2C%2C%22.%2C
    Request Version: HTTP/1.1
    Accept-Encoding: gzip,deflate\r\n
    Host: 10.124.211.96\r\n
    Accept: */*\r\n
    User-Agent: sqlmap/1.1.4#stable (http://sqlmap.org)\r\n
    Connection: close\r\n
    Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://10.124.211.96/newsdetails.php?id=1%27%2C..%2C%2C%2C%22.%2C]
  [HTTP request 1/1]
  [Response in frame: 109]

```

Here we can clearly see the URL encoding and the User-Agent:

In the last packet we see a long Query or SQLi payload:

We found out that the IP doing this SQLi is 10.124.211.200 and its internal IP. So for us to find out is it an malicious Insider doing this or an attacker who compromised a system and from it trying to exploit this SQLi.